Lecture Plan:

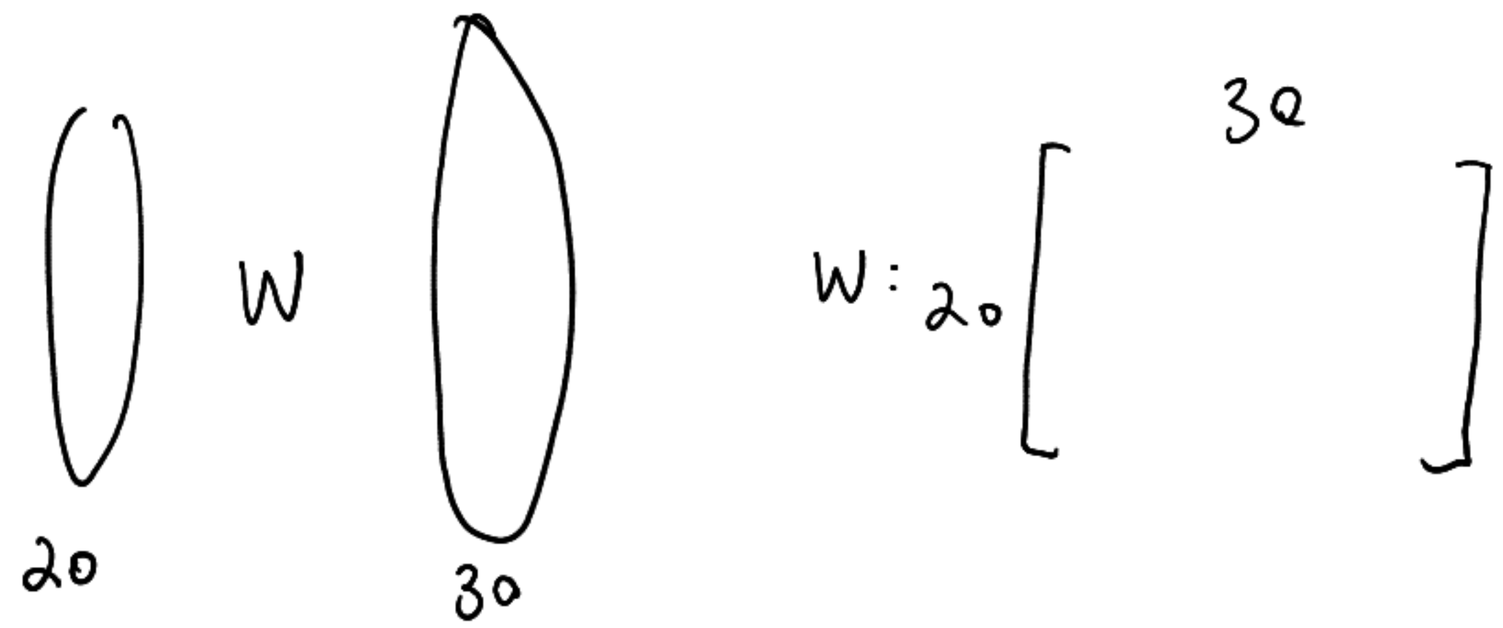
- Finish our discussion about ways to improve neural net learning

  * Weight initialization

  * Variants of stochastic gradient descent: momentum, Nesterov accelerated gradient, Adagrad, Adam, Asynchronous SGD

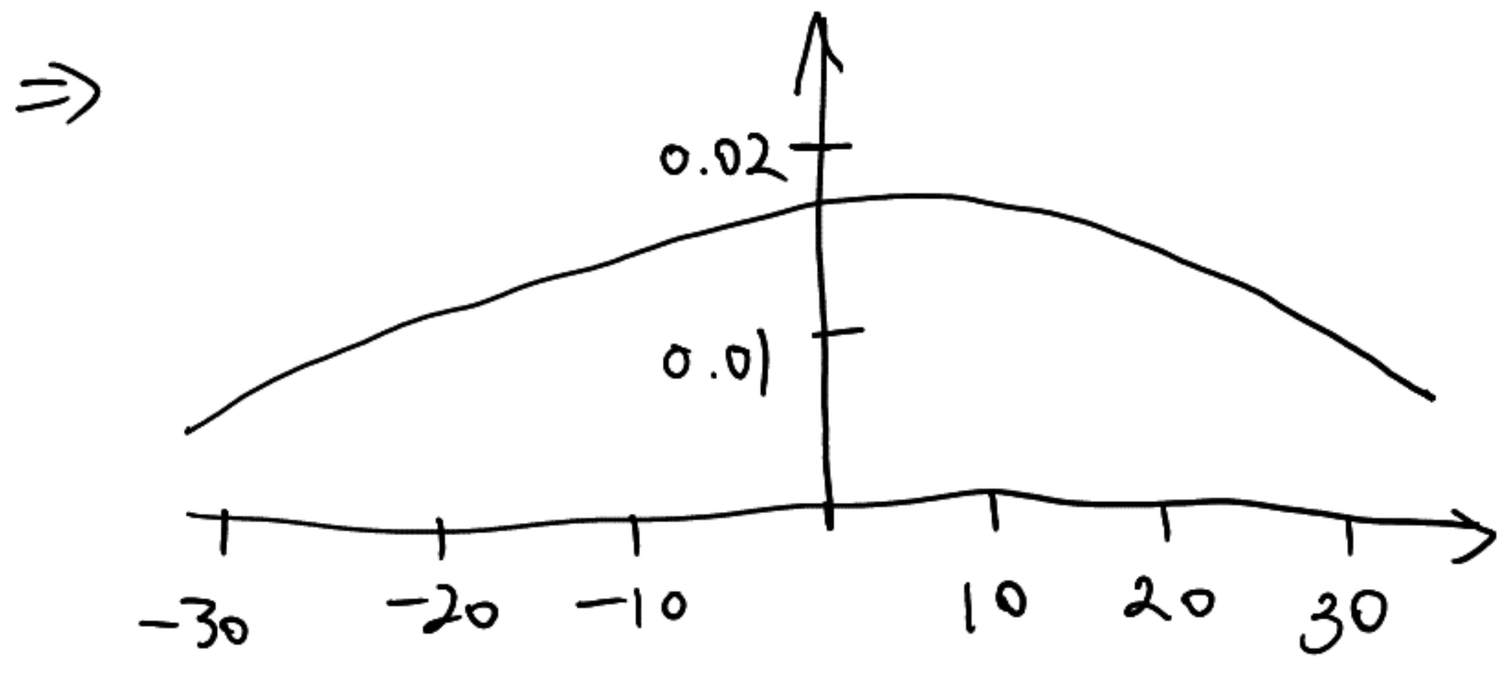
- Weight initialization

Example → Suppose we initialize our network using normalized Gaussians.



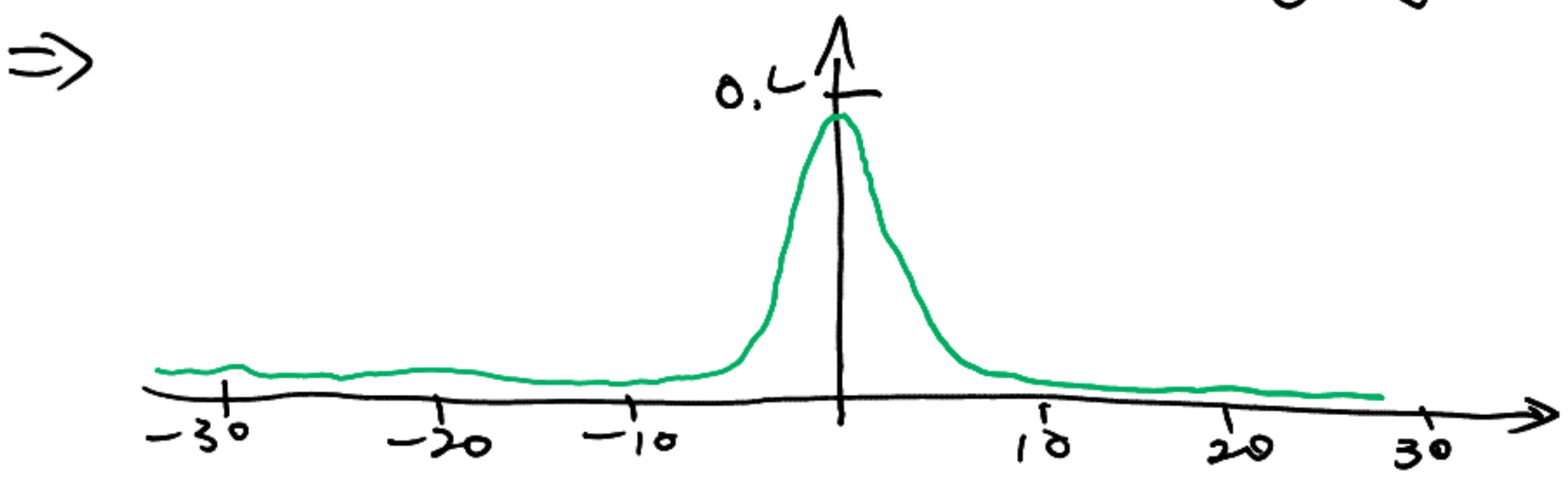
If $W_{i,j} \sim N(0,1)$, for any $1 \le i \le 20$, $1 \le j \le 30$,

then the norm of W is roughly $\approx \sqrt{20 \times 30} \approx 24.5$



Fix. Instead, let us sample $W_{i,j} \sim N(0, \frac{1}{20 \times 30})$,

then the norm of W is roughly ≈ 1



— Variants of Stochastic gradient decent

* Momentum-based gradient descent

Motivation: Hessian technique

Let's put neural networks aside for a bit. Instead, consider the abstrat problem of minimizing a function f. By Taylor expansion, $f(w)$ near a point w is approximated by

$$f(w+\Delta w) = f(w) + \sum_j \frac{\partial f}{\partial w_j}\Delta w_j + \frac{1}{2}\sum_{j,k}\Delta w_j \frac{\partial^2 f}{\partial w_j \partial w_k}\Delta w_k.$$

Write in more compact terms:

$$f(w+\Delta w) = f(w) + \boxed{\nabla f}\cdot \Delta w + \frac{1}{2}\Delta w^T \boxed{H}\Delta w + \cdots,$$

∇f: gradient vector

H: Hessian matrix

$$H_{jk} = \frac{\partial^2 f}{\partial w_j \partial w_k}$$

Suppose we approximate f by discarding the higher-order terms:

$$f(w+\Delta w) \approx f(w) + \nabla f\cdot \Delta w + \frac{1}{2}\Delta w^T H \Delta w.$$

Provided that the above is a good approximation of f, then the expression is minimized when

$$\Delta w = -H^{-1}\nabla f.$$

Therefore, we should expect that moving in the direction of Δw will significantly decrease the cost of f!

This suggests the following procedure for minimizing f.

- Choose an initialization w_0
- update $w^{(t+1)} \leftarrow w^{(t)} - \eta \cdot H^{-1}\nabla f$ for t iterations.

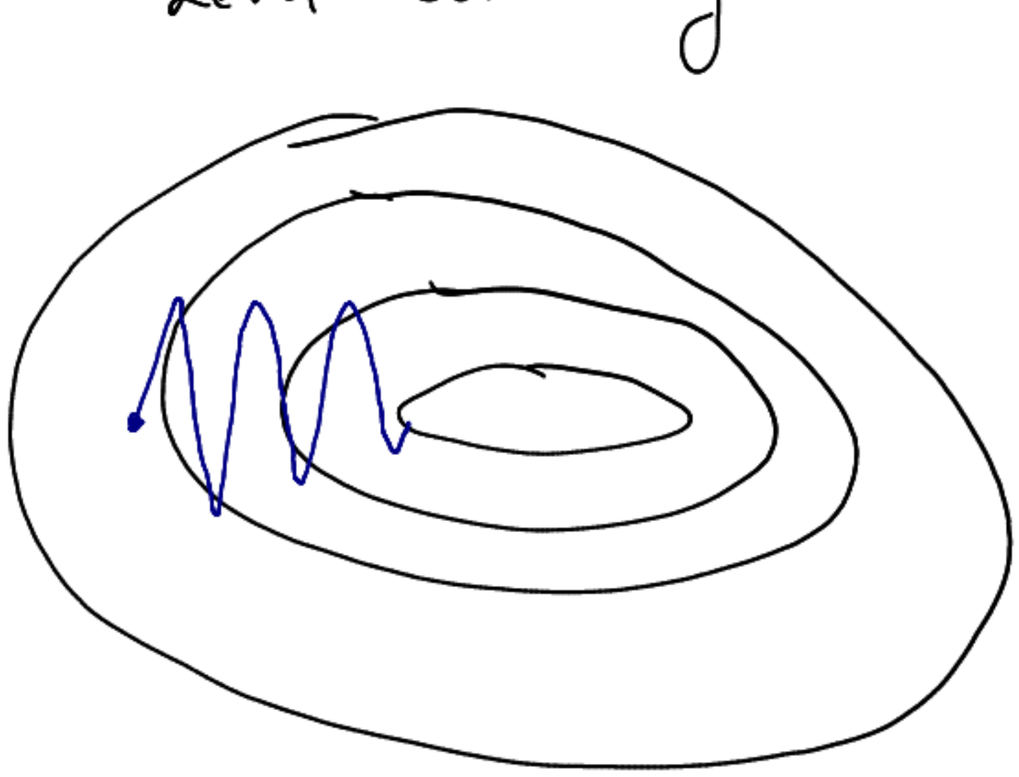
This approach to minimize f is known as the Hessian technique.

In convex optimization, Hessian technique is the most commonly used method!

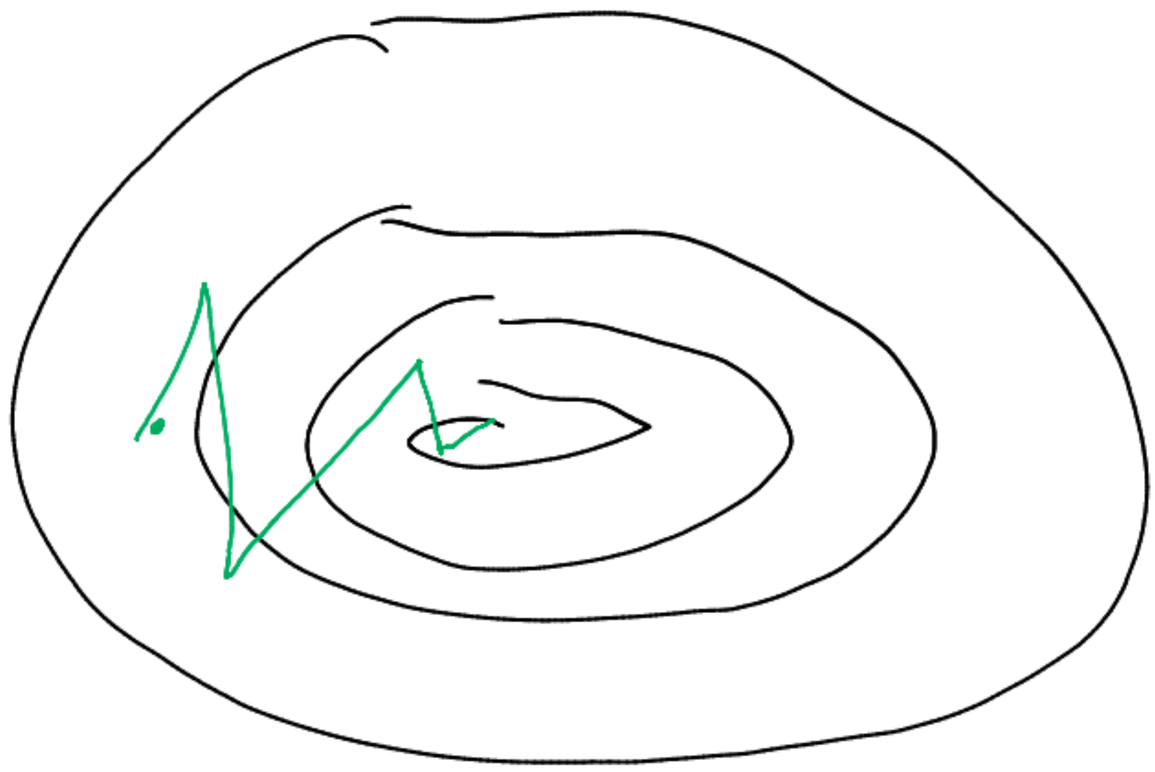
Theoretically, provided the f is convex and the Hessian has Bounded Lipschitz constant, then the Hessian technique converges in $O(|f(w^{(0)}) - c_{PT}|) + \log\log(\frac{1}{\varepsilon})$.

Momentum-based gradient descent based on a similar intuition, but avoids large matrices of second derivatives.

Level set diagram



SGD



SGD with momentum

$$v_t = \gamma \cdot v_{t-1} + \eta \cdot \nabla_w f(w)$$

$$w \leftarrow w - v_t$$

Two parameters here:

i) γ determines how much gradient information we want to take from prev iterations

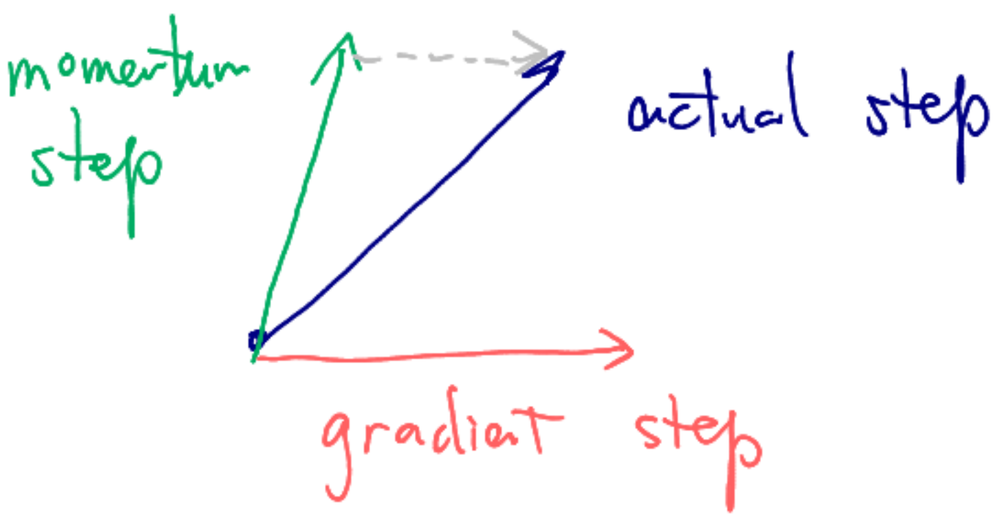
ii) η is the usual learning rate

Nesterov accelerated gradient

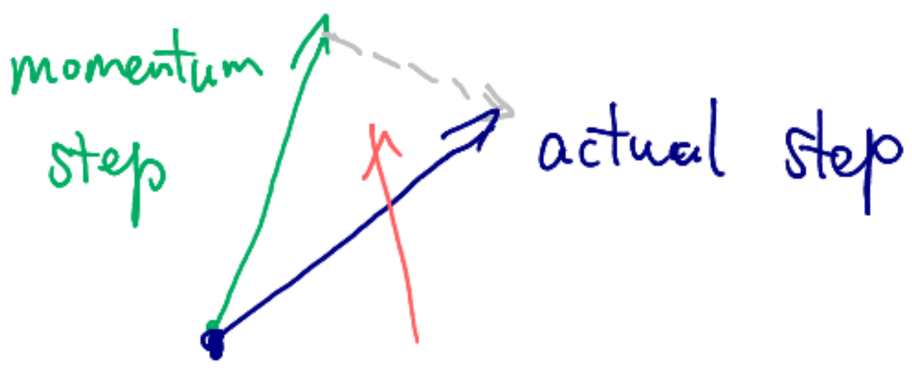
adds a correction term when calculating the gradient.

Observation: since we are going to change the parameter w by $\gamma \cdot v_{t-1}$. Therefore, for computing the gradient, we can treat the future approx position $w - \gamma \cdot v_{t-1}$ as a "lookahead". This is where our next update will be and so we will compute the gradient there.

Momentum update



Nesterov momentum update



"lookahead" gradient step (a bit different from original)

$$v_t = \gamma \cdot v_{t-1} + \eta \cdot \nabla_w f(w - \gamma \cdot v_{t-1})$$

$$w \leftarrow w - v_t$$

Adagrad : an adaptive learning rate scheme for gradient-based optimization.

Idea : Perform smaller updates (i.e. low learning rate) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rate) for parameters associated with infrequent features.

In SGD, we use the same learning rate η for every parameter : let $g = \nabla f(w)$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta \cdot g_i^{(t)}$$

In Adagrad, we modify the general learning rate η at every step t

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \frac{\eta}{\sqrt{G_{i,i}^{(t)} + \varepsilon}} \cdot g_i^{(t)}$$

ε → a small perturbation

$G_{ii}^{(t)}$ is a diagonal matrix where the (i,i)-th entry is the sum of the squares of the gradients w.r.t. w_i up to time t.